NATIONAL UNIVERSITY OF SINGAPORE

CS1010 – PROGRAMMING METHODOLOGY

(Semester 1: AY2015/16)

Time Allowed: 2 Hours

INSTRUCTIONS TO STUDENTS

1. This assessment paper consists of **THIRTEEN** questions and comprises **TEN** printed pages.
2. This is an **OPEN BOOK** assessment.
3. Calculators and electronic dictionaries are not allowed.
4. Answer all questions, and write your answers in the **ANSWER SHEETS** provided.
5. Fill in your Student Number with a pen, clearly on every odd-numbered page of your ANSWER SHEETS.
6. You may use **2B pencil** to write your programs.
7. Note that penalty will be given for programs that are unclear or unnecessarily long.
8. You must submit only the ANSWER SHEETS and no other document.

Section A: Multiple Choice Questions (MCQs) [12 marks]

There are six MCQs in this section. Each MCQ has one correct answer and is worth 2 marks. There is no penalty for wrong answer.

Q1. Based on the following function prototype and code fragment, which of the following correctly calls func()?

```
double func(int, double, int *, int []);

int a = 1, *b = &a, arr[5] = {0}, arr2[5][3] = {{0}};
double c = 1.0;
```

A. `func(*b, a, arr, arr2[1]);`
B. `int d = func(a, c, b, arr2);`
C. `double e = func(b, c, &arr[2], arr);`
D. `double f = func(*b, (double)a, arr2[0][0], arr2[2]);`
E. None of the above is correct.

Q2. Which of the following is true about this code fragment?

```
char *str1 = "app", *str2  = "pineapple";
char *str3 = strstr(str2, str1), str4[12]="pine";
strcat(str4, str3);
```

A. `str3` is a NULL pointer because the string "app" does not contains the string "pineapple".
B. `str3` cannot be used as the second argument for strcat().
C. `str4` contains the string "pineapp" at the end of the code fragment.
D. `str4` contains the string "pineapple" at the end of the code fragment.
E. None of the above is true.

Q3. Which of the following generates random real numbers in [-4, 4]?

A. `rand() % 8 - 4`
B. `rand() % 8.0 - 4`
C. `rand()/RAND_MAX * 8.0 - 4`
D. `(double)rand()/RAND_MAX * 8 - 4`
E. `(double)(rand()/RAND_MAX * 8 - 4)`

Q4. Given the following structure type definition, which of the following code fragment initializes the structure variable **student** to represent a student whose name, matric number and date of birth are "John", "A123456Y" and 1-Jan-1999 respectively?

```
typedef struct {
    int year;
    int month;
    int day;
} date_t;

typedef struct {
    char name[50];
    char matric[10];
    date_t dob;
} student_t;
```

(i)
```
student_t student = {"John", "A123456Y"};
student.dob.year = 1999;
student.dob.month = 1;
student.dob.day = 1;
```

(ii)
```
student_t student = {"John", "A123456Y", {1999,1,1}};
```

(iii)
```
student_t student = {"", "", {1999,1,1}};
student_t student_ptr = &student;
strcpy(student_ptr->name, "John");
strcat(student_ptr->matric, "A123456Y");
```

(iv)
```
student_t student;
student_t student2 = {"John","A123456Y", {1999, 1}};
student = student2;
student2.dob.day = 1;
```

A. (i) and (ii) only
B. (i) and (iii) only
C. (ii) and (iii) only
D. All except (iv)
E. None of the above options is correct.

Q5. What is the result of `f(13)`?

```
int f(int n){
    switch (n%4){
        case 0: return n;
        case 2: return 2 * f(n/4);
        default: return f(n-1)+f(n+1);
    }
}
```

A. 12
B. 16
C. 20
D. 24
E. 0

Q6. Which of the following is true about the linear search algorithm introduced in the lecture notes?

A. It produces incorrect results when applied on a sorted array.
B. It takes at least 1 comparison to find a given key if it appears in the array.
C. It may take more than n comparisons (where n is the size of the array) if the given key does not appear in the array.
D. It is always slower (i.e., requires more comparisons) than binary search.
E. It cannot be implemented using recursion.

Section B: Short Structured Questions [14 marks]

Q7. Write the output of the following program. [5 marks]

```
int main(void){
    int a = 1, b = 2, c = 3;
    f(&a, &b, c);
    printf("%d %d %d\n", a, b, c);
    return 0;
}

void f(int *a, int *b, int c){
    g(b, &c, *a);
    g(&c, a, *b);
}

void g(int *a, int *b, int c){
    *b += *a + c;
    c = *a + *b;
}
```

Q8. Mark the positions of 1s in the 2D array **arr** at the end of this code fragment. [4 marks]

```
int arr[5][5] = {{1}, {0,1}, {0,0,1}};
int i, j, k;

for (k = 0; k < 3; k++)
    for (i = 0; i < 5; i++)
        for (j = 0; j < 5; j++)
          if (arr[i][j])
              arr[j][4-i] = arr[i][j];
```

Q9. For a string of letters to be in LaTeX case, the first, third, fifth, ..., characters are in upper case while the rest in lower case.

For example, MoOn is in LaTeX case while MOon and mOoN are not.

The following function converts a string **str** into LaTeX case. Fill in the missing parts (a)-(c) without changing the rest of the code.

You may assume that the given string consists of only letters. [5 marks]

```
void convert(char str[]){
    int i;

    for (i = 0; i < [   (a)   ]; i++)
        if (i%2)
          [      (b)      ];
        else
          [      (c)      ];
}
```

Section C: Problem Solving [54 marks]

Q10. SEA games [22 marks]

The following structures contain information about the medals won by a country in the recent SEA games.

```
typedef struct {
   char sport[21];        // Name of the sport
   char type;             // Type of the medal: 'G' = gold,
                          // 'S' = silver and 'B' = bronze.
} medal_t;

typedef struct {
   char code[4];          // Code name of the country
   int numMedals;         // Number of medals
   medal_t medals[201];   // Medals won by the country
} country_t;
```

(a) Write a function

```
int readResults(country_t countries[])
```

to read data from a text file called **results.in** into a country_t array called **countries** and return the number of countries read. The first line of the file contains the number of countries – at least 1 and at most 20. This is followed by the information about the medals won by each country.

You should check that the file can be opened properly and handle the error when the opening of file fails.

An example of **results.in** is shown on the right. There are 5 countries in the file; the first country is Singapore, which has won 3 medals: silver for swimming, bronze for diving and gold for marathon.

[8 marks]

```
5
SIN 3
Swimming S
Diving B
Marathon G
INA 2
Badminton G
Diving S
MAS 3
Badminton B
Swimming B
Diving G
THA 2
Badminton S
Marathon S
VIE 2
Swimming G
Marathon B
```

Q10. *(continue...)*

(b) Write a function

```
void sortCountries(country_t countries[], int num)
```

that sorts a list of countries by the number of medals they have won **in descending order**. If there are multiple countries which have won the same number of medals, the order among these countries does not matter.

Your function should take in an array of country_t variables **countries**, which represents the list of the countries to be sorted, as well as an integer **num**, which represent the number of countries in the list. It should perform the sorting directly on the input array **countries**. [6 marks]

(c) Write a function

```
void findWinners(country_t countries[], int num,
                 char sport[], char winners[])
```

that returns a string consisting of the code names of the countries who have won the gold, silver and bronze medals respectively for a given sport.

For example, if the list of countries is as given in the example file in part (a), and the given sport is Swimming, the function should return the string **VIE*SIN*MAS**.

The first two inputs of this function are the same as the ones in part (b). In addition, your function should take in a char array **sport,** which contains the name of the given sport, as well as another char array **winners**, which is used to return the string of code names.

You may assume that the size of **winners** is sufficient for storing the message. You may also assume that there is exactly one winner for each medal of the given sport.

[8 marks]

Q11. Treasure Hunt

An island has 25 **treasures** arranged in a 5x5 grid. Each treasure is associated with a row index, a column index, as well as a positive integer which is the **value of the treasure**.

For example, as shown in the diagram on the right, the value of treasure (0,0) is 1, while the value of treasure (4,1) is 4.

	[0]	[1]	[2]	[3]	[4]
[0]	1	3	3	1	1
[1]	1	6	1	2	1
[2]	2	1	8	1	5
[3]	1	1	8	1	5
[4]	1	4	1	1	4

Captain Hook has recently obtained a magical machine for digging out these treasures but it only works on rectangular **areas** of a particular size. Such areas are denoted by the coordinates of the treasures at their upper left and lower right corners. These pairs of treasure coordinates are called the **coordinates** of the areas.

For example, area (0,0) – (0,0) consists of treasure (0,0) only. As another example, area (2,2) – (4,3) consists of treasures (2,2), (2,3), (3,2), (3,3), (4,2) and (4,3).

The **size of the area** is the number of treasures in the area, while the **total value of the area** is the sum of the values of the treasures in the area.

For example, the size and total value of area (0,0) – (0,0) are both 1. As for area (2,2) – (4,3), the size is 6, while the total value is 8+1+8+1+1+1 = 20.

Since the magical machine will vanish upon use, Captain Hook is interested in finding out which is the most valuable area of a particular size.

For example, the most valuable area of size 1 is either area (2,2) – (2,2) or area (3,2) – (3,2), whose total value is 8. In contrast, the most valuable area of size 6 is area (2,2) – (3,4), whose price is 28.

A function `int sumTreasures(int [][5], int, int, int, int)` is given to you which computes the total value of an area based on the treasure values and the coordinates of the area. For example, if **treasures** is a 2D array containing treasure values, you may call `sumTreasures(treasures, 1, 1, 2, 3)` to compute the total value of area (1,1) – (2,3).

Write a function `treasureHunt()` which takes in a 2D array of treasure values, and an integer representing the size of the area the magical machine can work on. It returns coordinates of the most valuable area of that particular size, as well as the total value of the area. If there is more than one most valuable area of the same total value, any one of them can be returned.

You are to determine the appropriate parameters and return type for this function.

[12 marks]

Q12. Strong Password Generator

In today's world, it is advisable to use strong passwords to protect your accounts from being hacked or stolen. Usually, such passwords are long and contain a mix of different types of characters.

A simple two-step process to generate a strong password is as follows:

1. Generate a string of 8 characters in the form of @@@@##%%, where each @ is a random lowercase letter, # a random uppercase letter and % a random digit. (These letters and digits need not be unique.)
2. Shuffle the characters in the string randomly.

For example, we can first generate a string fzheDP28 and then shuffle it to get 2Dehz8fP as a strong password.

Write a function

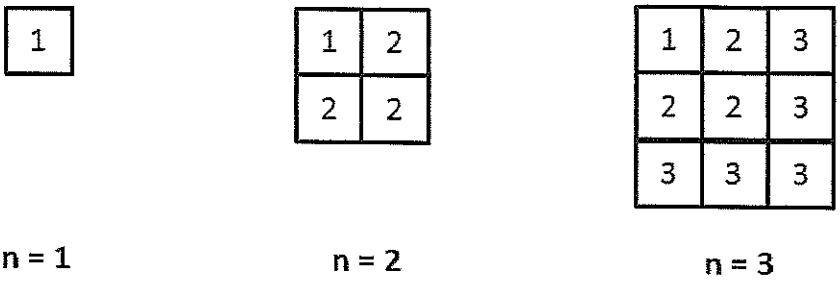
```
void generate(char password[])
```

to generate a strong password using this two-step process and store it as a string into the char array **password**.

(Hint: You should have learnt from the lecture on characters and strings that for a given digit character d, d – '0' gives you the numeric value of d (e.g., '8' – '0' = 8). You may want to reverse this process to generate the random digits and letters.) [12 marks]

Q13. Fill Pattern

Observe the pattern in the following 2D arrays.



1

n = 1

1	2
2	2

n = 2

1	2	3
2	2	3
3	3	3

n = 3

1	2	3	4
2	2	3	4
3	3	3	4
4	4	4	4

n = 4

1	2	3	4	5
2	2	3	4	5
3	3	3	4	5
4	4	4	4	5
5	5	5	5	5

n = 5

Write a recursive function

```
fill(int arr[][20], int n)
```

to fill a 2D array **arr** given **n** (1 <= **n** <= 20) with the pattern shown above.

No marks will be given if the function is not recursive. [8 marks]

=== END OF PAPER ===